Working Paper No. 11

As of 3/17/2018

Industrialization of the API

Interchangeable Parts

The essence of industrialization is the application of network effects to stuff.

There was a time when a screw-nut pair was an idiosyncratic artifact made by one craftsman to work as a matched pair, and there was no interchangeability.

Manufacturing became industrialized when the “interchangeable parts” meme kicked in. There is an interesting history around this. The popular myth in the US revolves around musket manufacturing for the Civil War.[1]

More relevant to this story is the history of screw threads.[2]

Interface Standards

In my view, these histories show that the 800-pound gorilla is not “interchangeable parts” but

**** Widespread adoption of interface standards ****

Industrialization applies beyond manufacturing. The most impressive example in our industry is the Internet Protocol, an application of industrialization to digital telecommunication.[3] Seen as a social phenomenon, at its root the story of the Internet is the story of the power of

**** Widespread adoption of interface standards ****

APIs and Screw Threads

The standardization of programming languages is industrialization of coding. Now, organizations are discovering the importance of applying network effects to the services they offer over the Internet. They are using the term “API” for their technical approach to this effort.[4]

Today, APIs are at the developmental stage of the idiosyncratic screw-nut pairs made by a single craftsman. *The parallel between APIs and fasteners is striking*, but the API equivalent of SAE screw-thread standards is nowhere to be seen.

---

[1] https://www.history.com/topics/inventions/interchangeable-parts

[2] https://en.wikipedia.org/wiki/Screw_thread#History_of_standardization

[3] https://en.wikipedia.org/wiki/Internet_Protocol

[4] Do a search on “API importance”. Here is one example: https://swaggerhub.com/blog/api-design/the-importance-of-standardized-api-design/

Industrializing APIs

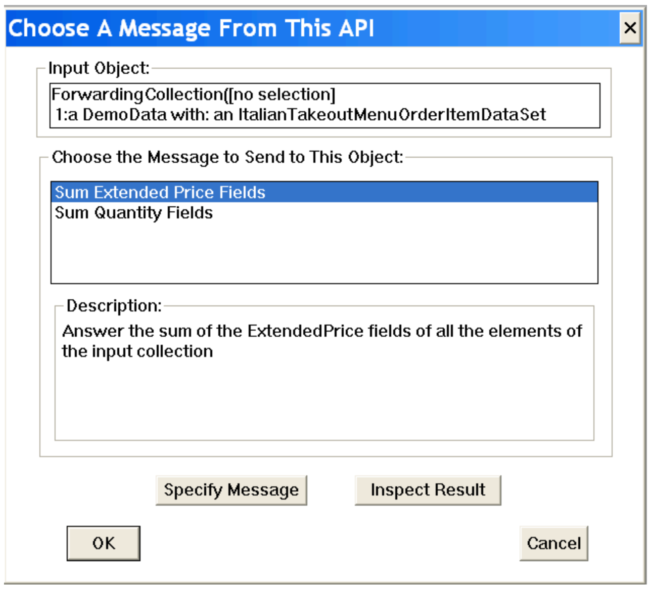
I see a way to do something about that. Here are two ways to describe the purpose of this effort.

- Maximize accessibility, that is, minimize demands on the knowledge of the user to the idiosyncracies of the API.
- Provide a standard interface that enables access to all APIs that conform to the standard, so that an API can be accessed and used in any appropriate programming context with minimum knowledge.

This can be the API analog of the SAE screw-thread standard. In the ideal, if the standard is defined and implemented well and widely adopted it can make APIs that conform to the standard accessible to *any programming tool* without requiring that the programmer have *a priori* knowledge of the peculiarities of the specific API.

An Approach

Here is one approach that employs my belief that the user interface of the standard must not rely on linear grammar. Give each logical group of services a URL, available to anybody. Within your IDE you can send a browser to that URL and you get a display something like this "Choose a Message" dialog.[5]



Choose A Message From This API

Input Object:
ForwardingCollection([no selection]
1:a DemoData with: an ItalianTakeoutMenuOrderItemDataSet

Choose the Message to Send to This Object:
Sum Extended Price Fields
Sum Quantity Fields

Description:
Answer the sum of the ExtendedPrice fields of all the elements of the input collection

[5] An adaption from http://melconway.com/Working/WP_10.pdf

In this case you are asking: What messages can I send to this collection of order items in my shopping cart applet? The list box is telling you that you can get the sum of all the extended price fields (unit price x quantity) of the items in the collection, or you can get the total number of individual items in the order.

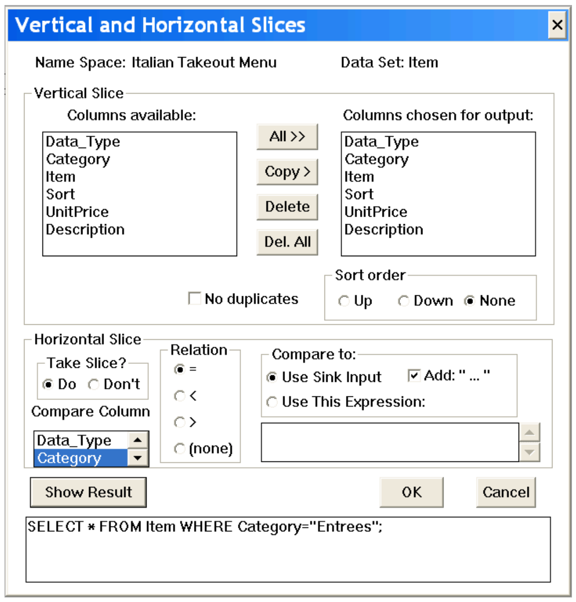
The list box gives you a list of brief, suggestive descriptions of messages (or entry points, in procedural terms). You can select any one and you get a more elaborate description of what it does. At that point you have a choice.

1. You can choose to be instructed about how it works and how to use it (this is not shown here; it is analogous to the *learn more* button found on websites), or

2. You can choose to connect the software you are building to it (the OK button does this).

If you choose to connect what happens at that point depends on the API, but the principle will hold that you will be led through a process, at every stage of which the "self-revealing" principle applies.[6]

If, for example, you want to perform a Select on a relational table, rather than writing an SQL statement you could click the "Specify Message" button and you might get something like this "Slice" dialog.



Vertical and Horizontal Slices

Name Space: Italian Takeout Menu   Data Set: Item

Vertical Slice

Columns available:		Columns chosen for output:
Data_Type	All >>	Data_Type
Category	Copy >	Category
Item	Delete	Item
Sort	Del. All	Sort
UnitPrice		UnitPrice
Description		Description

☐ No duplicates

Sort order: ◯ Up ◯ Down [✓ None]

Horizontal Slice

Take Slice? [✓ Do] ◯ Don't

Compare Column: Data_Type, [✓ Category]

Relation: [✓ =] ◯ < ◯ > ◯ (none)

Compare to: [✓ Use Sink Input] ☑ Add: " ... " ◯ Use This Expression:

```
SELECT * FROM Item WHERE Category="Entrees";
```

[6] http://melconway.com/Home/pdf/humanedozen.pdf page 3

The standard can support you in different ways.

1. You can test the connection using your data before committing to incorporating it into the software you are building (the “Inspect Result” button).

2. If you commit to connecting, the whole decision sequence will be saved as part of the connection so it can be revisited and amended, in the same form of presentation as at the time of examination and commitment (in other words, not as a script, for example).[7]

Consequences

The longer-term impact of the Industrialization of APIs remains to be discovered. I am exploring how this standard might restructure the open-source software community process to include non-programmers.[8]

Network effects have a history of being surprising. If the power of standardization is analogous between APIs and industrialization precedents such as screw-threads and digital telecommunications, the impact of API Industrialization could be larger than any of us can imagine.

[7] There are developers who will say: “too verbose”. This needs to be heard, so there might have to be alternative presentation styles. But “self-revealing” must remain as the default in order to be non-exclusionary, and the full graphical connection information must be available even if a shortcut is used in the process of making the connection. Without this, future maintenance of the connection by other persons would not derive the benefits of the standard.

[8] https://twitter.com/conways_law/status/972890099435364352 part 6-7